The 2nd ProgNova Multi-University Programming Contest

Editorial

Oct 15, 2016

A Seven Wonders

Count the number of occurrences of each letter (‘`T`’, ‘`C`’, ‘`G`’) and add their squares to the answer. Then add the minimum occurrence multiplied by 7 to the answer.

B Board Coloring

For any possible final configuration, there must exist a last step that creates a square of 3×3 cells with a same color. Any colors of those cells resulted from previous steps are overwritten. Therefore we may assume these cells previously have any colors from ‘`R`’, ‘`G`’, ‘`B`’. We label those cells with ‘`?`’, and then repeat finding such 3×3 squares, while allowing ‘`?`’ to represent any color. Note that white (‘`W`’) cannot occur in any chose square. When no more such squares can be found, the given final configuration is possible iff there exists no ‘`R`’, ‘`G`’, or ‘`B`’ on the board.

C Powers and Modulus

For any integer x between $[0, a)$, $x^b + (a-x)^b \equiv 0$ when b is odd. Therefore pairs of powers x^b and $(a-x)^b$ may cancel. When a is even, the answer is $(a/2)^b$, otherwise the answer is 0. It is possible to further observe that when $b > 1$ and $4 \mid a$, $(a/2)^b = (a/2)(a/2)^{b-1} \equiv 0$ since $(a/2)^b$ is even. When $b > 1$ and $2 \mid a$ but $4 \nmid a$, $(a/2)^b \equiv a/2$ since $(a/2)^{b-1}$ is odd. When $b = 1$ and a is even, the expression is $(a/2)(a+1) \equiv a/2$.

D Treasure Hunt

Run a shortest path algorithm that uses the distance pair (min number of days, max number of remaining stamina points) for each cell (i, j). Minimizing the first value of the pair is prioritized than maximizing the second value of the pair.

E Big Data

First pre-compute the prime numbers and the revenue for each subset of data pieces. Then do dynamic programming to solve $f(S)$, where S is the subset of unsold data pieces. We will pick a subset of S, denoted by $S' \subseteq S$. Then $f(S) = \max_{S'}\{f(S \backslash S') + revenue(S')\}$. The total time complexity is $O(3^N)$ for the DP plus the time to factorize the numbers.

F Exactly Paired

We use a slicing window. Suppose the window's right endpoint is now at index t. We want to pick the minimum left endpoint index s so that each element in the range $[s, t]$ appears exactly twice. For each number x, it may occur multiple times and give us a segment from which we can choose our left endpoint s. For example, if there are three x's to the left of t, we can illustrate it as `...x{...x}...x{...}j`. The segments within the curly brackets are the valid choices for our left endpoint t. We want to find the intersection of those segments for every number x, and then take the leftmost index in the intersection. We can use a segment tree to maintain those ranges. Adding 1 to every element inside a segment marks those elements to be invalid left endpoints in terms of one number x. Similarly, deducting 1 marks the elements valid in terms of one number x. We can maintain the sums on the positions inside a segment tree. By additionally storing the minimum value in the segment tree nodes, we can find the leftmost index that contains a zero in $O(\log N)$ time. When we increment j while moving the sliding window, we update and maintain the segments in the segment tree. The complete algorithm runs in $O(N \log N)$.

G Economical Coverage

Treat the grid as a chessboard with black and white cells. Create a bipartite flow graph in which each black node is bidirectionally connected with its adjacent white nodes. Each bidirectional edge has capacity K. The source is connected to every black node with a capacity of the black node's router cost. Each white node is connected to the sink with a capacity of the white node's router cost. The minimum cut of this flow graph gives the answer. The interpretation of this graph is that: Black nodes by default install routers. White nodes by default do not install routers. A node belonging to the source component indicates that we should toggle the decision for this node (install $\rightarrow$ not install, and vice versa). A node belonging to the sink component indicates that we should not toggle the decision for this node.

H Pokeball Fever

Let $f(n, b)$ be the expected cost to encounter n Pokemon with b balls in the beginning. We have

$$f(n,b) = \begin{cases} P \cdot f(n-1, b-1) + (1-P) \cdot f(n, b-1) & \text{if } b > 1 \\ f(n-1, 100) + 5 & \text{if } b = 0 \end{cases}$$

Each term $f(n, i)$ linearly depends on $f(n-1, j)$. We can create a matrix that encodes the coefficients of the contributions of $f(n-1, j)$ to $f(n, i)$ for every (i, j) pair. Note that to support implementation of adding 5 for the branch $b = 0$ we need to append a constant value 1 to the vector of $f(n, b)$. Therefore the vector is of size 102 (the number of balls has range $[0, 100]$, plus a constant 1). Consequently the coefficient matrix has size 102×102. Using efficient matrix exponentiation we can compute $f(N, 100)$ in $O(102^3 \log N)$ time.